

**SCHEDULING SYNCHRONIZATION OF PROGRAMS RUNNING AS STREAMS  
ON MULTIPLE PROCESSORS**

**Field**

5       The present invention relates scheduling in computer systems, and more particularly to synchronizing the scheduling of programs running as streams on multiple processors.

**Related Files**

This application is related to U.S. Patent Application No. \_\_\_\_\_, entitled  
10 "SCHEDULING SYNCHRONIZATION OF PROGRAMS RUNNING AS STREAMS ON  
MULTIPLE PROCESSORS", filed on even date herewith; U.S. Patent Application No.  
\_\_\_\_\_, entitled "Multistream Processing System and Method", filed on even  
date herewith; to U.S. Patent Application No. \_\_\_\_\_, entitled "System and  
Method for Synchronizing Memory Transfers", Serial No. \_\_\_\_\_, filed on even  
15 date herewith; to U.S. Patent Application No. \_\_\_\_\_, entitled "Decoupled Store  
Address and Data in a Multiprocessor System", filed on even date herewith; to U.S. Patent  
Application No. \_\_\_\_\_, entitled "Decoupled Vector Architecture", filed on even  
date herewith; to U.S. Patent Application No. \_\_\_\_\_, entitled "Latency Tolerant  
Distributed Shared Memory Multiprocessor Computer", filed on even date herewith; to U.S.  
20 Patent Application No. \_\_\_\_\_, entitled "Relaxed Memory Consistency Model",  
filed on even date herewith; to U.S. Patent Application No. \_\_\_\_\_, entitled  
"Remote Translation Mechanism for a Multinode System", filed on even date herewith; and to  
U.S. Patent Application No. \_\_\_\_\_, entitled "Method and Apparatus for Local  
25 Synchronizations in a Vector Processor System", filed on even date herewith, each of which is  
incorporated herein by reference.

**Copyright Notice/Permission**

A portion of the disclosure of this patent document contains material that is subject to  
copyright protection. The copyright owner has no objection to the facsimile reproduction by

anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever. The following notice applies to the software and data as described below and in the drawings hereto: Copyright © 2003, Cray, Inc. All Rights Reserved.

5

### Background

Through all the changes that have occurred since the beginning of the computer age, there has been one constant, the need for speed. In general, this need has been satisfied in one or both of two methods. The first method involves making the hardware faster. For example, 10 each new generation of hardware, be it processors, disks, memory systems, network systems or bus architectures is typically faster than the preceding generation. Unfortunately, developing faster hardware is expensive, and there are physical limitations to how fast a certain architecture can be made to run.

The second method involves performing tasks simultaneously through parallel 15 processing. In parallel processing, two or more processors execute portions of a software application simultaneously. Parallel processing can be particularly advantageous when a problem can be broken into multiple pieces that have few interdependencies.

While parallel processing has resulted in faster systems, certain problems arise in 20 parallel processing architectures. One problem that arises is that the parallel processors often share resources, and contention for these shared resources must be managed. A second problem is that events affecting the application may occur and one or more of the parallel processes may need to be informed of the event. For example, an exception event may occur when an invalid arithmetic operation occurs. Each parallel processing unit of an application may need to know of the exception.

25 As a result, there is a need in the art for the present invention.

## Summary

The above-mentioned shortcomings, disadvantages and problems are addressed by the present invention, which will be understood by reading and studying the following specification.

5 One aspect of the system and method is that a process is stated in an operating system. Additionally, a plurality of program units associated with the process are started. When a context shifting event occurs, each of the plurality of program units has their scheduling synchronized and their context set so that each thread processes the context shifting event.

10 A further aspect of the system is that some program units may be executing on more than one multiple processor unit. In the operating system selects a multiple processor unit to host all of the program units, and migrates those program units that are not currently on the selected multiple processor unit to the selected multiple processor unit.

15 The present invention describes systems, clients, servers, methods, and computer-readable media of varying scope. In addition to the aspects and advantages of the present invention described in this summary, further aspects and advantages of the invention will become apparent by reference to the drawings and by reading the detailed description that follows.

## Brief Description Of The Drawings

20 FIG. 1 is a block diagram of parallel processing hardware and operating environment in which different embodiments of the invention can be practiced;  
FIG. 2 is a block diagram of a parallel processing software environment according to embodiments of the invention; and  
FIG. 3 is a flowchart illustrating a method according to an embodiment of the invention.

25

### **Detailed Description**

In the following detailed description of exemplary embodiments of the invention, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific exemplary embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical, electrical and other changes may be made without departing from the scope of the present invention.

Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the ways used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar computing device, that manipulates and transforms data represented as physical (e.g., electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

In the Figures, the same reference number is used throughout to refer to an identical component which appears in multiple Figures. Signals and connections may be referred to by

the same reference number or label, and the actual meaning will be clear from its use in the context of the description. Further, the same base reference number (e.g. 120) is used in the specification and figures when generically referring to the actions or characteristics of a group of identical components. A numeric index introduced by a decimal point (e.g. 120.1) is used  
5 when a specific component among the group of identical components performs an action or has a characteristic.

The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

10

### Operating Environment

FIG. 1 is a block diagram of parallel processing hardware and operating environment 100 in which different embodiments of the invention can be practiced. In some embodiments, environment 100 comprises a node 101 which includes two or more multiple processor units 102. Although two multiple processor units 102.1 and 102.2 are shown in FIG. 1, it will be  
15 appreciated by those of skill in the art that other number of multiple processor units may be incorporated in environment 100 and in configurations other than in a node 101. In some embodiments of the invention, node 101 may include up to four multiple processor units 102. Each of the multiple processor units 102 on node 101 has access to node memory 108. In some embodiments, node 101 is a single printed circuit board and node memory 108  
20 comprises daughter cards insertable on the circuit board.

In some embodiments, a multiple processor unit 102 includes four processors 104.1 – 104.4 and four cache memory controllers 106. Although each multiple processor unit is shown in FIG. 1 as having four processors, those of skill in the art will appreciate that other embodiments of the invention may have more or fewer processors 104. In some  
25 embodiments, each processor 104 incorporates scalar processing logic (S) and vector processing logic (V). In some embodiments, each cache memory control 106 may access 512 KB of memory. Each of processor 104 may access any one or more of the cache memory controllers 106.

In one embodiment, the hardware environment is included within the Cray X1 computer system, which represents the convergence of the Cray T3E and the traditional Cray parallel vector processors. The X1 is a highly scalable, cache coherent, shared-memory multiprocessor that uses powerful vector processors as its building blocks, and implements a modernized vector instruction set. In these embodiments, multiple processor unit 102 is a Multi-streaming processor (MSP). It is to be noted that FIG. 1 illustrates only one example of a hardware environment, and other environments (for other embodiments) may also be used.

FIG. 2 is a block diagram of a parallel processing software environment 200 according to embodiments of the invention. In some embodiments, software environment 200 comprises an operating system that manages the execution of applications 202. Applications may also be referred to as processes. In some embodiments of the invention, the operating system is a UNIX based operating system, such as the Unicos/mp operating system from Cray Inc. However, the invention is not limited to a particular operating system.

Application 202 may be configured to run as multiple program units. In some embodiments, a program unit comprises a thread 206. Typically, each thread 206 may be executed in parallel. In some embodiments, an application may have up to four threads and the operating environment assigns each thread to be executed on a different processor 104 of multiple processor unit 102. In some embodiments, the threads 206 of an application may be distributed across more than one multiple processor unit 102. For example, two threads 206.1 and 206.2 of an application 202 may be assigned to processors 104.1 and 104.2 respectively while another two threads 206.3 and 206.4 may be assigned to processors 104.7 and 104.8 respectively.

In some embodiments, each application 202 has an application context 204 and each thread 206 has a thread context 208. Application context 204 and thread context 208 are used by the operating environment 200 to manage the state of an application and thread, and may be used to save and restore the application state as the application or thread is moved on or off a processor 104. In some embodiments, application context 204 includes information such as the memory associated with the application, file information regarding open files and other operating system information associated with the application. Thread context 208 includes

information such as the register state for the thread, a signal state for the thread and a thread identification. The signal state includes information such as what signals are currently being handled by the thread and what signals are pending for the thread.

5 Certain events require synchronization among the threads running as part of an application. For example, an event requiring a context shift for the application or thread may occur, and other threads running as part of the application may need to be informed or may need to handle the event.

10 FIG. 3 is a flowchart illustrating a method for scheduling multiple threads in a parallel processing environment according to an embodiment of the invention. The method to be performed by the operating environment constitutes computer programs made up of computer-executable instructions. Describing the methods by reference to a flowchart enables one skilled in the art to develop such programs including such instructions to carry out the methods on suitable computers (the processor or processors of the computer executing the 15 instructions from computer-readable media). The method illustrated in FIG. 3 is inclusive of acts that may be taken by an operating environment executing an exemplary embodiment of the invention.

20 The method begins when an application is started within an operating system (block 310). Typically the application will be scheduled on one of the processors in the system as one of many processes executing within an operating environment.

25 Next, the application indicates that threads should be started (block 320). The operating system arranges for the threads to be scheduled on one of the available processors. During the execution of one or more of the threads, a context shifting event may occur (block 330). There are multiple reasons for context shift events, the quantity and type of context shifting event will depend on the operating environment.

In some embodiments of the inventions, the context shifting event is a “signal.” A signal in Unicos/mp and other UNIX variations is typically an indication that some type of exceptional event has occurred. Examples of such events include floating point exceptions when an invalid floating point operation is attempted, a memory access exception when a

process or thread attempts to access memory that does not exist or is not mapped to the process. Other types of signals are possible and known to those of skill in the art.

Additionally, it should be noted that in some operating environments, a signal may be referred to as an exception.

5        In alternative embodiments, the context shifting event may be a non-local goto. For example, in Unicos/mp and other UNIX variants, a combination of “setjmp()” and “longjmp()” function calls can establish a non-local goto. In essence, the “setjmp” call establishes the location to go to, and the “longjmp” call causes process or thread to branch to the location. The goto is a non-local goto because it causes the execution of the thread or  
10      process to continue at a point outside of the scope of the currently executing function. A context shift is required, because the processor registers must be set to reflect the new process or thread execution location.

In further alternative embodiments, the context shifting event may be a system call. Typically a system call requires that the process or thread enter a privileged mode in order to  
15      execute the system call. In Unicos/mp and UNIX variants, the system call must typically execute in kernel mode, while normally a process or thread executes in user mode. In order to execute in kernel mode, a context shift is required.

Those of skill in the art will appreciate that other context shifting events are possible and within the scope of the invention.

20       Upon receiving indication of a context shifting event, the operating environment must synchronize the state of the threads associated with the application (block 340) In some embodiments of the invention, this synchronization may be performed by a “g sync” instruction. The operation of the g sync instruction is described in detail in United States Patent Application Serial No. \_\_\_\_\_ entitled “System and Method for  
25      Synchronizing Memory Transfers” which has been previously incorporated by reference.

Additionally, in some embodiments, the operating system determines if each of the threads are operating on the same multiple processor unit 102. In these embodiments, the system will select a multiple processor unit 102 that is to manage all of the threads, and swaps

out threads that are executing on the selected multiple processor unit 102 that do not belong to the application associated with the context shifting event.

The operating environment then migrates threads associated with the application that are currently not on the selected multiple processor unit 102 to the selected multiple processor unit (block 344). This is desirable because often the threads will be executing code located in the memory of the selected multiple processor unit. For example, in the case of a signal, the threads will each execute the same signal handling code established for the signal.

Next, each thread's context is set so that it executes in the new context (block 350). For example, in the case of a signal, the new context will be the signal handler code. In the case of a system call, the new context will typically be code executed in kernel mode.

As each thread executes in the new context, local synchronization may be required, for example when a thread leaves a system call or when one thread requires results calculated by another thread. In some embodiments of the invention, an “msync” instruction may be used to synchronize the execution of the threads. The operation of the msync instruction is described in detail in United States Patent Application Serial No. \_\_\_\_\_ entitled “System and Method for Synchronizing Memory Transfers” which has been previously incorporated by reference.

### Conclusion

Systems and methods for scheduling threads in a parallel processing environment have been disclosed. The systems and methods described provide advantages over previous systems. The threads associated with a process and executing in parallel may be conveniently scheduled so that they all process a context shifting event.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiments shown. This application is intended to cover any adaptations or variations of the present invention.

The terminology used in this application is meant to include all of these environments. It is to be understood that the above description is intended to be illustrative, and not

restrictive. Many other embodiments will be apparent to those of skill in the art upon reviewing the above description. Therefore, it is manifestly intended that this invention be limited only by the following claims and equivalents thereof.